ISSUE: JANUARY 1998 LISTing Policy ..... Annual Dues $16.00

LISTings

Newsletter of the Long Island Sinclair / Timex Users' Group

15 YEARS AND STILL GOING STRONG!

Pres.	Bob Gilder pro tem
Vice Pres.	Bob Gilder
Tresurer	Robert Malloy
Cor. Secy.	John Pazmino
Assoc. Editors	Fred Stern
Publisher	Bob Gilder
Libr.	Tom Skapinski

Please send all inquiries and submissions (including dues) to: L.I.S.T.
Mr. Robert Gilder
69 Jefferson Place
Massapequa, N. Y. 11758

COMING EVENTS: The next L.I.S.T. meeting will be Sunday, 2/8,98 at 2 P.M. at the home of Bob Malloy, 412 Pacific Street Massapequa Pk , New York 11762
Tel: 516-541-6731

NEXT MEETING: FEBRUARY 8, 1998

On e sample copy sent upon receipt of business size SASE. Copies provided on Exchange basis with other Bona fide user groups. We are always looking for articles, programs, reviews, etc to keep members informed and entertained. You maintain full credit and copyright.

READ, DATA & RESTORE

If you wish to use data from within a program rather than input values from a keyboard, which for a long list may tend to be very tedious, we can use the READ-DATA statements. For example, suppose we wish to incorporate a list of prices of several items in a program, we can utilize READ-DATA statements as follows:

```
100 READ Hammercost,Nailcost,Nailcost,Drivercost,Screwcost
110 data 2.50,0.01,1.75.0.02
```

The values of the variables Hamercost,Nailcost,Drivercost,Screwcost in the READ statement 100 are assigned the values in the exact corresponding order as given in the DATA statement of 110. So after execution of Line 100, data would have been taken from the DATA statement and assigned as follows:

Hamercost = 2.50,Nailcost = 0.01,Drivercost =1.75,Screwcost = 0.02

Each READ statement in a program consists of READ followed by a list of variable identifiers, each identifier being separated from the next by a comma. Each DATA statement is a list of values and/or expressions, each value mey be put anywhere you wish in a program - normally at the beginning or at the end if this is convenient - since the computer ignores any DATA statements until it meets a READ statement.

The first time the computer, in executing the program meets a READ statement it takes for the first variable value variable and the first data value from the DATA list, for the second variable and the second data value, and so on, working its way through the whole of the DATA statement lists.

Note: In order to use READ successfully on the QL, before it is used for the first time in the program, you must use the RESTORE command so that the READ and DATA statements are aligned.

If the number of items in the READ and DATA statements do not match, e.g. if there are more variables in the READ statements that values in the DATA statements, then the computer will display an error message immediately as it tries to READ a vale which is not there.

1 This simple program illustrates the basic function of READ-DATA statements.

```
  5 CLS: RESTORE
 10 READ A,B,C
 20 PRINT"FIRST ITEM A= ";A
 30 PRINT"SECOND ITEM B=";B
 40 PRINT"THIRD ITEM C=" ;C
 50 DATA 111,222,333

 RUN<ENTER>

FIRST ITEM A= 111
SECOND ITEM B=222
THIRD  ITEM C=333..the display obtained on screen showing that A is
assigned the first value in the DATA statement list, B the second and C
the third.
```

Note: Line 5 contains two instructions, CLS to clear the screen, and the RESTORE command to 'restore" or bring the data pointer to the beginning of the DATA list, so that it definitely points at the first value.

2 This program works out the average value of a number of items listed in READ-DATA statements.

```
 5 CLS: RESTORE
10 REMark ** To find the average of a list of values **
20 READ A,B,C,D,E,F,G
30 READ H,I,J,K,L,M,N
40 SUM= A+B+C+D+E+F+G
50 SUM= SUM+H+I+J+K+L+M+N
60 PRINT "AVERAGE=";SUM/14
70 DATA 14,23,32,47,56,66,72,83
80 DATA 93,104,116,127,138,147

RUN <ENTER>
AVERAGE=79.85714...Display of results obtained
```

3 This program obtains DATA using READ-DATA statements and works out sub-total costs and total costs for tools and materials purchased:

30 hammers at $2.75 each
2,800 nails at $0.005 each
14 screwdrivers at $1.75 each
1,585 screws at $0.01 each
20 sheets of hardboard at $12.25 each

```
 5 CLS:RESTORE
10 REMark ** COST PROGRAM **
20 READ No_HAM,HAM_Cost
30 T_H_Cost=No_HAM*HAM_Cost
40 READ No_Nail,Nail_Cost
50 T_N_Cost=No_Nail*Nail_Cost
60 READ No_Driv,Driv_Cost
70 T_D_Cost=No_Driv*Driv_Cost
80 READ No_Screw,Screw_Cost
90 T_S_Cost=No_Screw*Screw_Cost
100 READ NO_Hardb,Hardb_Cost
110 T_HB_Cost=NO_Hardb*Hardb_Cost
120 Total_Cost=T_H_Cost+T_N_Cost+T_D_Cost
130 Total_Cost=Total_Cost+T_N_Cost+T_HB_Cost
140 PRINT"Total Hammer Cost =";T_H_Cost
150 PRINT"Total Nail Cost=";T_N_Cost
160 PRINT"Total Screw-Dr.Cost=";T_D_Cost
170 PRINT"Total Screw Cost=";T_S_Cost
180 PRINT"Total Hardbd Cost=";T_HB_Cost
190 PRINT"- - - - - - - - -  - - - - - -"
200 PRINT"Total Cost=";Total_Cost
210 DATA 30,2.75,2800,5E-3,14,1.75
220 DATA 1585,1E-2,20,12.25

RUN <ENTER>
Total Hammer Cost  = 82.50
Total Nail Cost    = 14.00
Total Screw-Dr Cost= 24.50
Total Screw Cost   = 15.85
Total HardBd Cost  =245.00
-------------------------
Total Cost         =381.85 ... display obtained
```

Note: The total will NOT be 381.85 because the QL changed two elements in both DATA lines, with exponent numbers. Total is 380.

Frequently we may wish to STOP program execution at a given line and make a check on the results so far obtained before allowing the execution to continue. This is especially useful in checking intermediate results in a lengthy program.

For STOPing execution of a program at a given line, SuperBASIC provides the STOP statement:

```
100 STOP
```

This statement STOPs execution of the program at line 100 and will return SuperBASIC to the command mode.

The CONTINUE command allows a program which has been interrupted by a STOP statement (or by a BREAK action) to be resumed at the next line after the STOP statement. (To BREAK program execution , hold down the CTRL key and press the space bar. This will return the cursor to the screen.) For example, consider the program:

```
 5 CLS: RESTORE
10 DATA 70,450,12.3,67,90,23.6
20 DATA 34.8,923,65.4,768.9
30 READ A,B
40 PRINT (A-32)*5/9,B*4.2
50 STOP
60 READ C,D,E
70 PRINT (C+D)*A-E
80 STOP

RUN <ENTER>

21.111111    1890
CONTINUE <ENTER>
5461
```

On running the program, execution starts from the beginning, READs in DATA for A,B at line 30, outputs results at line 40 and then is stopped by the STOP statement at line 50. Program execution can be restarted from line 60 by typing in the direct mode command CONTINUE. A further READ is made at line 60, results output at line 70 and then execution stopped at line 80 by a second STOP statement.

In 'normal' use of READ-DATA statements, the computer works from the beginning of the first DATA statement (the one with the lowest line number) when it meets the first READ statement and then subsequently works works through all the DATA lists until it reaches the end.

This sequence can be 'restored' by using RESTORE statements which have the form:

```
100 RESTORE
100 RESTORE (line number)
```

The first RESTORE statement (with no line numbers after RESTORE) forces any subsequent READ statement to start reading data from the first value of the first DATA statement (lowest number) in the program. When a specified line number is given after RESTORE, then any subsequent READ statement takes its data from the first DATA statement at or immediately following the specified line number in the RESTORE statement.

For example, let us work through the following program:

```
 5 CLS:RESTORE
10 READ A,B,C
20 PRINT A,B,C
30 RESTORE
40 READ X,Y,Z
50 PRINT X,Y,Z
60 READ A,B,C
70 PRINT A,B,C
80 DATA 1,2,3,4,5,6

RUN <ENTER>

1    2    3   ...(display of A,B,C at line 20)
1    2    3   ...(display of X,Y,Z at line 50)
4    5    6   ...(display of A,B,C at line 70)
```

At line 10 the READ statement assigns A=1, B=2 and C=3, i.e. the first three values in the DATA list of line 80. Thus at line 20 we obtain the display:

```
1    2    3
```

Line 30, the RESTORE statement, returns us to the beginning of the DATA list. Thus at line 40, the READ statement assigns X=1, Y=2, Z=3 and line 50 produces display (again):

```
1    2    3
```

AT line 60, however, the READ statement 'pointer' is at the fourth item in the DATA list and hence this READ statement assigns A=4, B=5, C=6 and this is confirmed by the action of the display statement 70:

```
4    5    6
```

Finally, consider a second example:

```
5 CLS:RESTORE
10 DATA 1,2,3,4,5,6
20 DATA 10,20,30,40
30 DATA 100,200,300
40 READ A,B,C
50 PRINT A,B,C,(A+B+C)/3
60 RESTORE 20
70 READ A,B,C,D
80 PRINT A,B,C,D,(A+B+C+D)/4
90 RESTORE 10
100 READ A,B,C,D,E
110 READ F,G,H,I,J,K,L,M
120 PRINT A;B;C;D;E;F;G;H;I;J;K;L;M

RUN <ENTER>

1    2    3    2
10   20   30   40    25
1234561020 30 40 100 200 300
```

is the display obtained.
At line 60 the RESTORE statement restores the READ pointer to the first value in DATA statement 20. Thus the READ statement at line 70 assigns

A=10, B=20, C=30, D=40. In the absence of 60 RESTORE 20, DATA would have been taken from the next value in line 10, i.e. A=4, B=5, C=6 and then from line 20, D=10.

AT line 90, the RESTORE 10 statement returns the READ pointer to the beginning of DATA list from the first value in 10 and continuing with 120 to work right through the complete DATA statements 10, 20 and 30.

The next listing is a simple telephone directory using READ, RESTORE and DATA statements.

```
5 REMark LIST friends
10 REMark Data statement example
15 CLS
100 DIM name$(10,28)
110 RESTORE
120 :
130 FOR count = 1 TO 7: READ name$(count)
140 PRINT name$
150 DATA "Bob Malloy    516-541-6731 ", "Tom Skapinsky 516-732-1825",
"Fred Stern    561-852-6899", "John Pazmino  212_337_2618", "George G
ilder 718-544-1106", "Robert Gilder 516-541-2271", "Joe LaPunzina 718
-373-5022"
```

The output:

```
Bob Malloy     516-541-6731
Tom Skapinsky 516-732-1825
Fred Stern     561-852-6899
John Pazmino   212-337-2618
George Gilder 718-544-1106
Robert Gilder 516-541-2271
Joe LaPunzina 718-373-5022
```

In order to make the output display uniform, spaces were padded between the Names and Telephone numbers.

```
5 CLS
7 RESTORE
10 REMark ** Exchange Rates **
20 DIM country$(10,15)
30 DIM EX_RATE(10)
40 FOR j=1 TO 10
50 READ country$(j)
60 PRINT"Enter Exchange Rate"
70 PRINT "`1 for ";country$(j); "=";
75 INPUT EX_RATE(j)
80 END FOR j
90 PRINT "List of exchange rates"
100 FOR j=1 TO 10
110 PRINT country$(j),EX_RATE(j)
120 END FOR j
130 DATA "Austria   ","Belgium   ","Denmark   ","France   "
140 DATA "W.Germany","Greece     ","Italy     ","Holland     "
150 DATA "Switzerland","U.S.A.     "
```

Exchange Rates utilizes both string and numeric variable arrays. The string information - names of countries - is held in DATA statements. The input statement in conjunction with the FOR statement is used to ENTER in numerical data - exchange rates, in this example. The final part of the program displays the countries and the corresponding exchange rate. See you next month... *Bob Gilder*

Howdy T/S 2068 fans! I hope you have all been well. I must apologize for this article being late. I am the Editor for our users group newsletter, the RAMTOP. Let me tell you, that it takes a lot of time! I think at least one-quarter of my time must be trying to convince my wife that I won't be much longer on the computer. Oh well, on to the matter at hand. Last time I promised you an article and a program about the TONE DIALER. Here it is.

So many of you feel that the SOUND command is a real mystery. I'm no pro at it but I will try to help you out with it. Your 2068 has two methods for producing audio to the internal speaker.

1 - The BEEP command. This uses the SCLD chip (which is the chip that controls the video, cassette I/O, keyboard etc).

2 - The sound command. This uses the BASIC of your 2068 to send data to a special chip known as a Programable Sound Generator. In the case of the 2068 it is a General Instruments AY-3-8912. Within this chip there are three independent tone generators and one noise generator. There are three outputs that are mixed to one of the speaker. By sending a series of numbers to this chip we can control these tones and noise generators. To do this, we must learn how to use the SOUND command. You should refer to chapter 21 of your user manual. I will outline it for you.

There are 15 registers in this chip. For the sound we can use 0 - 13. On page 192 you will see just what each register is for. Some of these are a bit confusing. I have found that the course and fine tune are not too hard to figure out if you realize just what they are doing. The fine tune register is an eight (8) bit number and the course is a 4 bit number, togrther they are for a 12 bit number. (The upper 4 bits of the course tunes arent used). This is where it gets rather complicated.

As you know, the 2068s memory is divided up into 8 bit bytes. This allows a figure of 0-255 to be stored. If we put 2 bytes together we can form a 16 bit byte. This allows a number of 0-65535 to be stored. In the case of the sound chip, it uses an 8 bit byte for the fine tune, (0-255) but it uses a 4 byte for the course tune. (0-15). The real trick is that to figure just what the frequency that you want for any of the tone generators to produce. You must first divide the frequency that you want INTO the Master clock frequency of the sound chip.

That number is: 110,250 Hz. (Hz = Hertz, the unit of measure for frequency). Let's say you want to produce a tone of about 1,000 Hz. First we divide 110,250 by 1000. It is also less than 255 so the course tune will be 0. The actual frequency that the sound chip will produce will be: 1002.27. As you can see, most frequencies you want won't be exact. Now let's see how to tell the 2068 to "play" a 1K(kill or 1000) tone.

First we get the word SOUND by getting the "E" cursor, then hold the Symbol Shift key and press the "g' key. Now we must tell the 2068 which tone (or noise) generator we want to activate. This is done with

registor #7. Let's keep it simple for now and just use A for tone. That will mean that registor 7 will be set to 62 (63-1). Now, we tell the 2068 what the course and fine tune will be for channel A. For A we use registor Ø for fine and 1 for course. Since course is Ø we need not worry with registor 1 but we must set Ø to 11Ø. Now finally we must tell the 2068 how loud to make the tone. You may sdjust the amplitude (volume) of three channels independently.

In our case we will want to use channel A which is register 8. The levels are Ø (off) to 15 (loudest). You better make it loud since the speaker is so small. You nowNow use register 7=62, Ø=11Ø, and 8=15. The line format would be: 1Ø SOUND register 7=62;Ø,11Ø;8,15. If you were to "RUN" this you won't hear anything because the 2Ø68 shuts of the sound chip upon completing a program or an immediate command. So, to play the note for 1 second, enter a line 2Ø as such: 2Ø PAUSE 6Ø. You may combine them to form chords or use the noise generator to make sound effects. To get into sound effects I would suggest getting Sam's Intermediate/Advanced Guide.

If you study the Tone Dialer program, you will see how I got the frequencies needed. (lines 1ØØØ- 1Ø11). Each button on a tone phone produces 2 tones mixed. The trick was to find out just what the frequencies were. I finally got them at the library. The program is easy to use. Most of it is self explanatory. Once you type it in, RUN it. If all is well, when you press a number key you will hear the tones for that number. The # and * are produced by pressing their keys, along with Symbol/Shift. After you press a few numbers try yhr "r" key. If all is well, it should produce the same numbers again in rapid succession. (REDIAL!). For those numbers that require a pause such as dialing from an office where you must dial a 9 first to get the dial tone, enter a "p" at that point. The pause is about 1.5 seconds. If you need more just add another "p". To dial another number press the "c" key to clear the redial memory and dial the new number. Using the Memory to store and dial (with redial) up to 2Ø numbers is also easy. Simply follow the prompts. When dialing from the memory, the only thing you must remember, is to press the "c" key to go back to the menu or to dial a different number. You may have up to 3Ø digits (including pauses) in each of the 2Ø memories.

When the memory list is displayed, only the first 16 digits are displayed due to screen space. This is really the best for the long distance services that you must have True Tone. I have used this program many a time to call long distance with Allnet. If you don't have the cable and amplifier as in my last article (AUG CTM) you will have to put the phone mic right at the speaker of the 2Ø68. I just place the phone next to the speaker of my monitor. Well, have fun!

If you have any problems or if you like this, please write! I won't continue to write if no one is interested. My address is:

James G. DuPuy
6514 Bradley Avenue,
Parma, OH 44129

Using a Tone Dialer the easy way. . .

By James G. DuPuy

```
   5 DIM a$(20,30): DIM b$(20,10
)
  10 REM TONE DIALER - J.G. DU
PUY '85 '86
  11 BORDER 6
  12 CLS : PRINT FLASH 1;"
T O N E   D I A L E R     "
  14 PRINT '''"0 - 9,#,*,p TO DIA
L MANUALY"'"R --- TO REDIAL LAST
 NUMBER"'"S --- TO STORE A NUMBE
R"'"M --- TO DIAL FROM MEMORY"'"
C --- TO CLEAR REDIAL"'"S AND CA
P SHIFT TO SAVE TO TAPE"'"  WITH
 THE NUMBERS IN THE MEMORY"'
  15 PRINT '"Entering a ""p"" wi
ll allow a"'"delay of about 1.5
seconds."'"When using Memory, if
 you get a busy, press ""r"" to
redial"'"or press ""c"" to go ba
ck to the"'"menu."
  18 LET r$=""
  19 POKE 23609,10
  20 LET n$=INKEY$: IF n$="" THE
N GO TO 20
  22 IF n$="m" THEN GO TO 2000
  24 IF n$="s" THEN GO TO 3000
  25 IF n$="p" THEN GO TO 32
  26 IF n$="r" THEN GO TO 4000
  27 IF n$="c" THEN GO TO 12
  28 IF N$="S" THEN GO TO 5000
  29 IF n$="*" OR n$="#" THEN GO
 TO 32
  30 IF n$>"9" OR n$<"0" THEN GO
 TO 20
  32 PRINT n$;"-";
  35 LET r$=r$+n$
  36 IF n$="#" THEN GO TO 1010
  37 IF n$="*" THEN GO TO 1011
  38 IF n$="p" THEN PAUSE 100: G
O TO 20
  40 GO TO VAL n$+1000
 100 SOUND 0,l;1,0;2,h;3,0;7,60;
8,15;9,15
 105 IF n$="r" OR n$="m" THEN PR
INT r$(i);"-";
 110 IF INKEY$<>"" THEN GO TO 11
0
 120 PAUSE 5
 130 SOUND 8,0;9,0;7,63
 134 IF n$="r" OR n$="m" THEN RE
TURN
 140 GO TO 20
1000 LET l=117: LET h=82: GO TO
100
1001 LET l=158: LET h=91: GO TO
100
1002 LET l=158: LET h=82: GO TO
100
1003 LET l=158: LET h=75: GO TO
100
1004 LET l=143: LET h=91: GO TO
100
1005 LET l=143: LET h=82: GO TO
100
1006 LET l=143: LET h=75: GO TO
100
1007 LET l=129: LET h=91: GO TO
100
1008 LET l=129: LET h=82: GO TO
100
1009 LET l=129: LET h=75: GO TO
100
1010 LET l=117: LET h=75: GO TO
100
1011 LET l=117: LET h=91: GO TO
100
```

```
2000 REM MEMORY CALL
2005 GO SUB 2010: GO TO 2100
2010 CLS : PRINT FLASH 1;"     M
 E M O R Y   L I S T       "
2015 LET v=0
2020 FOR j=1 TO 20
2030 PRINT AT j,0;j;"- ";b$(j);"
 - ";a$(j)
2035 IF a$(j)="
                " THEN LET v=v+1
2040 NEXT j
2050 RETURN
2100 IF v=20 THEN PRINT "NO NUMB
ERS STORED YET.": PAUSE 120: GO
TO 12
2110 INPUT "WHICH # TO DIAL? ";
Q: IF Q>20 OR Q<1 THEN GO TO 211
0
2120 IF A$(Q)="                " THEN
 PRINT "NO # AT ";Q: PAUSE 100:
GO TO 12
2130 PRINT B$(Q);" - ";
2140 LET R$=A$(Q)
2150 GO TO 4005
3000 REM MEMORY STORE
3010 GO SUB 2010
3020 INPUT "WHICH # TO STORE? ";
Q: IF Q>20 OR Q<1 THEN GO TO 302
0
3030 PRINT Q;
3040 INPUT "NAME? "; LINE B$(Q)
: IF b$(q)="                " THEN GO
TO 3040
3045 PRINT " - ";B$(Q);" - ";
3050 INPUT "PHONE #? "; LINE Z$
: IF LEN Z$>30 OR LEN z$<1 THEN
GO TO 3050
3055 LET A$(Q)=Z$
3060 PRINT A$(Q)
```

```
3070 INPUT "C-CORRECT, ENT-CONT.
";X$: IF X$="c" THEN GO TO 3030
3075 GO SUB 2010: INPUT "E-M-MEN
U, S- TO STORE A #   :";x$: IF x$
="s" THEN GO TO 3020
3080 GO TO 12
4000 REM REDIAL
4002 IF INKEY$<>"" THEN GO TO 40
02
4003 PRINT "REDIAL ";
4005 FOR i=1 TO LEN r$
4008 IF r$(i)=" " THEN GO TO 402
0
4010 IF r$(i)="#" THEN GO SUB 10
10: GO TO 4020
4012 IF r$(i)="*" THEN GO SUB 10
11: GO TO 4020
4013 IF r$(i)="p" THEN PAUSE 100
: GO TO 4020
4015 GO SUB 1000+VAL r$(i)
4020 NEXT i
4025 PRINT
4030 GO TO 20
5000 REM SAVE TO TAPE
5005 CLS : PRINT FLASH 1;"ENTER
PROGRAM NAME.                YOU CA
N ONLY ENTER UP TO 10      CHARAC
TERS."
5010 INPUT x$: IF LEN x$>10 OR L
EN x$<1 THEN GO TO 5010
5015 CLS : PRINT AT 10,12; FLASH
 1;"SAVING"
5020 SAVE x$ LINE 12
5030 GO TO 12
```

James G. DuPuy
6514 Bradley Ave.(Down)
Parma, OH 44129 CTM

ATTENTION LIST Subscribers: When it is time to renew your membership, (look at your mailing label), please make out your check to Robert Gilder, LIST President or to Robert Malloy, Treasurer. PLEASE DO NOT MAKE OUT YOUR CHECK to LIST. Our bank requires a large amount of money in a savings account in order to cash checks. THANK YOU!

Robert Gilder
69 Jefferson Place,
Massapequa, NY 11758

Robert Malloy
412 Pacific Street,
Massapequa Park, NY 11762

Due to rising postage costs outside of the United States, we must raise our annual dues accordingly:

USA postage $16.00
CANADA and MEXICO $17.50 US, and the rest of the world $24.00 US.

Bob Malloy, LIST Treasurer

WHO'S ONLINE

Some of us here at LIST have been wondering how many of our members are using modems with their Sinclair computers. It would be helpful if those of you who are into communications would take a few minutes to let us have the following info.

COMPUTER USED .....
COMMS PRGRM .....
BAUD RATE .....
EMAIL ADDRESS.....
ONLINE SERVICES USED.....
SUGGESTIONS FOR LIST.....

You can reply to me at either of the following addresses:
74776.2342@compuserve.com
bmalloy@chelsea.ios.com (Internet)

Or, you can use our snailmail address.

Bob Malloy

ON LINE

Joe LaPunzina	Jpunzi@aol.com
Bob Malloy	bmaloy@idt.net
Tom Skapinski	tskapins@juno.com
John Pazmino	john.pazmino@moondog.com
Tim Swenson	swensotc@ss2.sews.wpfb.af.mil
Bill Cable	bcable@triton.coat.com
Mike Jonas	mjonas@bbn.com
Gary Norton	gnorton@world.std.com
Al Boehm	boehm@plh.af.mil
Ed Kingsley	elk4@aol.com
Ken Harbit	krh03@cvip.fresno.com